THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 1, January 1983
Printed in U.S.A.

The 3B20D Processor & DMERT Operating System:

3B20D Central Processing Unit

By M. W. ROLUND, J. T. BECKETT, and D. A. HARMS

(Manuscript received March 18, 1982)

The 3B20D Processor has been developed to meet the need for very reliable, real-time control of a variety of Bell System applications. To achieve its high-reliability goals, most of the major subsystems within the processor are duplicated, including the Central Processing Unit (CPU). The CPU uses a 32-bit architecture throughout, including the memory and input/output buses. Extensive self-checking logic is employed. The 3B20D CPU is microprogrammed to select dynamically up to four instruction sets. The microstore uses a 64-bit word with up to 16K words of high-speed bipolar PROM or RAM available. This rich emulation capability makes the 3B20D Processor ideal for emulating existing instruction sets and porting existing software. Peripheral units are connected to the CPU via the Direct Memory Access unit (DMA). The DMA controllers provide direct memory transfers between the main store and peripheral devices, reducing the load placed on the central control to process input/output requests.

I. CENTRAL CONTROL STRUCTURE

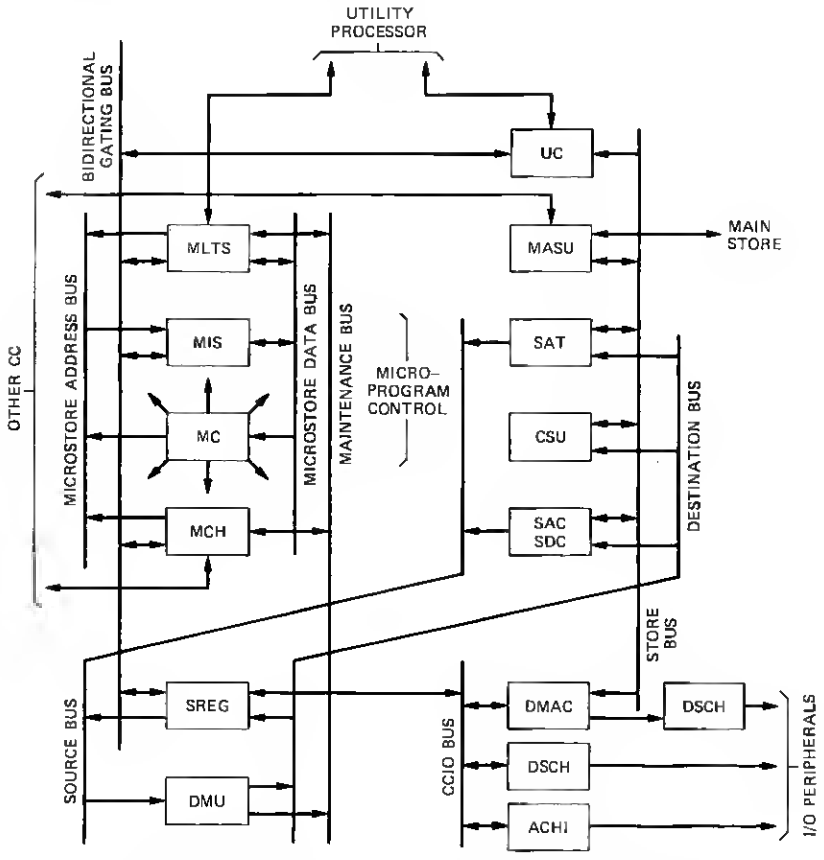
The 3B20D Processor performs all the functions normally associated with a CPU and several unique functions as well. The unique functions include the features necessary for reliable duplex operation, efficient emulation of other machines, and communication with a flexible and intelligent periphery. The processor employs an extensive microprogram capability to minimize the amount of hardwired decoding and to simplify the writing of microcode. There is substantial flexibility in the choice of instruction formats that can be efficiently interpreted.

The CPU is a 32-bit machine with a 24-bit address. Most of the data paths in the Central Control (CC) are 32 bits wide (plus 4 parity-check

bits). A fully self-checking philosophy is applied within each CPU without dependence upon matching between the duplex CPU's. The CC design is based on the register type of architecture, with multiple buses to allow concurrent data transfers. Separate I/O and store buses provide the capability of concurrent store and I/O operations.

1.1 Major subsystems

A detailed block diagram of the CC structure is shown in Fig. 1. The



ACHI – APPLICATION CHANNEL INTERFACE
CC – CENTRAL CONTROL
CCIO – CENTRAL CONTROL INPUT/OUTPUT
CSU – CACHE STORE UNIT
DMAC – DIRECT MEMORY ACCESS CONTROLLER
DMU – DATA MANIPULATION UNIT
DSCH – DUAL SERIAL CHANNEL
MASU – MAIN STORE UPDATE
MC – MICROCONTROL UNIT
MCH – MAINTENANCE CHANNEL
MIS – MICROINSTRUCTION STORE
MLTS – MICROLEVEL TEST SET
SAC – STORE ADDRESS CONTROLLER
SAT – STORE ADDRESS TRANSLATOR
SDC – STORE DATA CONTROLLER
SREG – SPECIAL REGISTERS
UC – UTILITY CIRCUIT

Fig. 1—The 3B20D Processor central control.

major subsystems and their associated functions are described in the following sections.

1.1.1 Microcontrol

This subsystem provides nearly all of the complex control and sequencing operations required to implement the instruction set. The microcode can support up to three different emulations in addition to its native instruction set. Other complicated sequencing functions are also stored in the microstore (MIS). The Microcontrol (MC) unit sequences the microstore and interprets each of its words to generate the needed control signals specified by the microinstruction. To optimize the execution of microinstructions, execution time depends upon the complexity of the microinstruction. Each is allocated sufficient time to implement the microinstruction in multiples of 50 nanoseconds. These times are 150, 200, 250, and 300 nanoseconds. The wide 64-bit word allows a sufficient number of independent fields within the microinstruction to perform a number of simultaneous operations. Some common and high-runner instructions are implemented with a single microinstruction.

1.1.2 Data manipulation unit

The arithmetic and logic operations are carried out in the Rotate Mask Unit (RMU) and the Arithmetic/Logic Unit (ALU). These two units are connected in series to comprise the Data Manipulation Unit (DMU) shown in Fig. 2. The RMU can rotate or shift any number of bit positions from 0 to 31 through a two-stage barrel-shift network. In addition, a selection of AND/OR operations can be performed on bits, nibbles (4 bits), bytes, half words, full words, and miscellaneous predefined patterns. The RMU outputs feed directly into the ALU. Any bit fields within a word can easily be manipulated and processed by the DMU, greatly enhancing the power of the microcode. The ALU is implemented using 2901 bipolar 4-bit processor elements (see Fig. 3).[1] Eight such chips provide two key elements: the 2-port (A and B), 16-word (RAM) and the high speed ALU. Data in any of the 16 words addressed by the 4-bit A-address-field input can be used as an operand to the ALU. Likewise, data in any of the 16 words defined by the 4-bit B-address-field input can be simultaneously read and used as a second operand to the ALU. The result can be directed to the RAM word specified by the B-field. To take advantage of the above feature, the internal 16-word RAM is dedicated as general registers. This enables the arithmetic and logical operation involving general registers and/or the output of the RMU to be performed at the optimum speed.

The logic blocks of Fig. 2 reveal the way the RMU-ALU is self-checked. The rotate unit is checked by word parity, which it preserves,

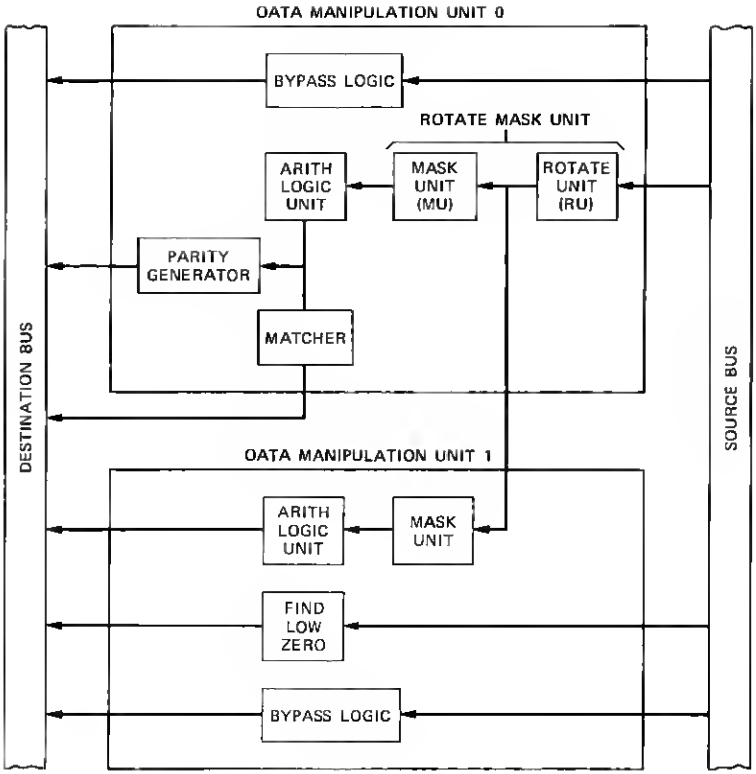


Fig. 2—Block diagram of the data manipulation unit.

and the mask unit is checked by duplication. The ALU is also duplicated. The data is taken from one ALU and parity is generated from the other. The data from one ALU is also matched with the duplicate. The underlying self-checking strategy, illustrated here and used throughout the CPU, is to use parity checking in those cases where parity is preserved and duplication of logic elements where parity is not preserved.

1.1.3 Special registers

The 16 general registers reside inside the DMU and are available to the programmer. A number of Special Registers (SREG) associated with the operation of the CC are external to the DMU. Most of them are not explicitly specified by the instruction. They are characterized by their special dedicated functions with additional inputs from sources other than the internal data bus. Their outputs are used to control and direct the operation of the processor. Some of the SREG's are: the Error Register, Program Status Word, the Hardware Status Register,

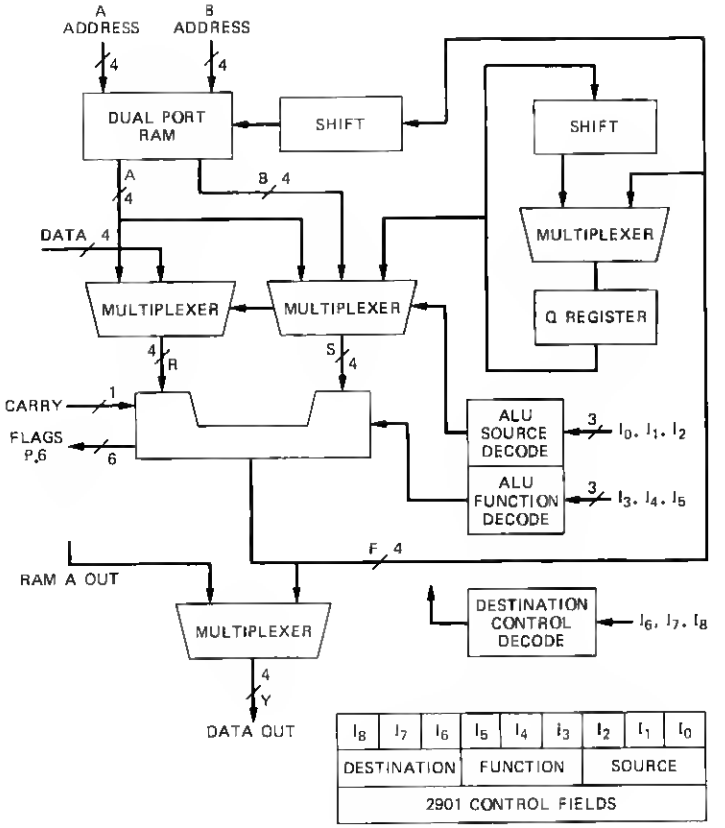


Fig. 3—The 2901 internal architecture.

the System Status Register, the Interrupt Register, timers, etc. In addition, a 32-word RAM is provided within the SREG block and is essentially available only at the microcode level. It is used for scratch-pad space and preassigned registers to facilitate and enhance the power of microprogram sequences. The registers supporting memory management[2] are an example of such preassigned registers.

1.1.4 Store interface control

The store interface circuit controls the transfer of data or instructions from the memory system[2] to the CC. Several SREGs are associated with the store interface. Associated with the Store Address Control (SAC) are the Program Address (PA), the Store Address Register (SAR), and the Store Control Register (SCR). Associated with the Store Data Control (SDC) are the Store Data Register (SDR), the Store Instruction Register (SIR), and the Instruction Buffer (IB).

The SAC and SDC together make up the store interface that handles the memory addressing, the updating of the program counter, and the fetching and prefetching of instructions. The circuit ensures a continuous flow of instructions to be interpreted by the microcontrol unit.

1.1.5 Store address translation

Memory mapping is required in the implementation of a virtual addressed multiprogramming system. The Store Address Translation (SAT) facility containing the Address Translation Buffer (ATB) is the mechanism that provides a mapping between a program-specific virtual address and its corresponding physical address. Address translation hardware[2] is provided to facilitate memory management in a more efficient manner. The store address space is divided into 128 segments, each having up to 64 pages, with a page containing 2K bytes. Both the virtual and the physical address spaces are 24 bits. The complete set of virtual-to-physical address translation tables are stored in the main memory. A significant amount of time would be required by the CC in the repetitive task of dynamic address translation in using the main store tables. This translation time is reduced substantially by storing the likely-to-be-used physical address translations in a high-speed cache-like Address Translation Buffer (ATB).

1.1.6 Main store update

The Main Store Update (MASU) unit[2] provides a multiport interface to the memories as both DMA and CC attempt to use the memory. The update circuit arbitrates asynchronous requests from the on-line CC, both DMAs, and the off-line CC. The cross coupling between the memory update units permits the on-line CPU to access either memory or both memories for concurrent write operations.

1.1.7 Input/output interface

The communication path between the CC and the I/O channels is through the CCIO bus. It is a local, high-speed, direct-coupled, parallel bus. Direct memory access between the main store and peripheral units is provided by a Direct Memory Access Controller (DMAC) that communicates with intelligent peripheral units via Dual Serial Channels (DSCH). I/O channels including user-specific interfaces can be connected directly to the CC via the CCIO bus. Two such standard interfaces are the DSCH, a high-speed multiport serial interface, and the Application Channel Interface (ACHI), a high-throughput, parallel bus, peripheral communication path.

1.1.8 Cache

The cache[2] is an optional circuit equipped to improve the overall

system performance by reducing the effective memory access time. The cache is a fourway set associative memory containing a total of 8K bytes.

1.1.9 Maintenance channel

This circuit provides diagnostic access to the CC at the microinstruction level. It also is used to control basic fault recovery and system sanity functions in the off-line processor.

1.2 Major buses

The processor structure diagrammed in Fig. 1 allows three key transfers to proceed simultaneously. The first is a microcontrol path through the Microinstruction Store (MIS). The second is a computation path through the Data Manipulation Unit (DMU). The third is a memory path through the cache. The CPU is pipelined at both the microstore and store levels: microinstruction execution is overlapped with microstore read and instruction execution is overlapped with fetch. In addition, separate transfer paths are provided for maintenance and input/output.

Microaddresses pass from the Microcontrol Unit (MC) to the MIS over the 16-bit Microstore Address (MSA) Bus. Microinstructions pass from the MIS to the MC over the 64-bit Microstore Data (MSD) Bus. Control signals derived from the microinstruction fan out to the subsystems that connect to the 32-bit Source (SRC) bus and 32-bit Destination (DST) bus.

The SRC and DST buses are the primary gating paths for computations. Data and addresses pass from the DMU to the SDC and SAC, respectively, over the DST bus. The SREG are also accessed over the SRC and DST buses. The Store Bus includes both a 32-bit data field and a 24-bit address field. The store can also be accessed by the DMAC over the same bus structure.

The Maintenance Channel (MCH) controls the processor using the MSA and MSD buses. The 32-bit Bidirectional gating bus (BGB) and Maintenance (MTC) bus are used by the MCH to observe the CPU. The BGB is also used to load the writeable microstore.

The CCIO bus is a bus linking the CC with the I/O channels. Operations over the bus are programmed I/O instructions. Control of the DMA circuits is exercised over this bus but the data transfers via DMA are routed directly onto the main store bus.

II. I/O FACILITIES

The I/O facilities[3,4] are designed to meet a wide range of applications with different needs and capabilities. A modular and flexible I/O communication structure is provided by means of dedicated point-to-

point channels. The loose coupling of the processor to the peripherals allows considerable freedom to grow and expand the system with minimum physical constraints. The I/O architecture is shown in Fig. 4. Two channels have been designed for the CCIO bus, the dual serial channel and the application channel interface.

2.1 Input/output channels

2.1.1 Dual serial channel

The Dual Serial Channel (DSCH) (see Fig. 5) is a modular circuit that provides high-speed serial links between the CC or DMA and the peripheral devices. Direct memory access is only supported on devices connected to DSCHs on a DMAC. Transfers for devices directly connected to the CCIO bus are completely controlled by CC software using specific I/O instructions.

The DSCH supports word (36 bits) or block (16 words) transfers at either of two rates. For peripheral devices located within 100 feet of the DSCH a 10-MHz clock rate is used. For devices up to 250 feet away a 5-MHz clock rate must be used.

The DSCH provides a private serial point-to-point data path to each of the 16 intelligent devices it supports. Each link uses a five twisted-pair cable as the transmission media. Two of the pairs are used for bidirectional data transmission, two pairs for unidirectional clocks and the fifth pair for the peripheral device to set flags in the DSCH. Data is transmitted using RS422 compatible signaling. The other end of the cable connects to a Duplex Dual-Serial Bus Selector (DDSBS) that converts the signals from the DSCH into a parallel format. Each DDSBS can interface to two DSCHs allowing the peripheral device to be connected to both CPU's of the duplex processor.

2.1.2 Application channel interface

The Application Channel Interface (ACHI) (see Fig. 6) provides a differential dc parallel bus and control signals for high-throughput programmed I/O. The circuit can only reside on the CCIO bus and has no autonomous sequencer. The ACHI's relies on command sequences from the CC to carry out peripheral operations.

The ACHI's CCIO bus interface is similar to that of the DSCH. The same data bus, addressing, control, and response signaling is used. The parallel bus generated on the peripheral side of the ACHI consists of two unidirectional 36-bit data buses and 11 control and response leads. One data bus is for input to the ACHI and one is for output with concurrent data transfers allowed.

2.2 Direct memory access controller

The Direct Memory Access Controller (DMAC) (see Fig. 7) provides

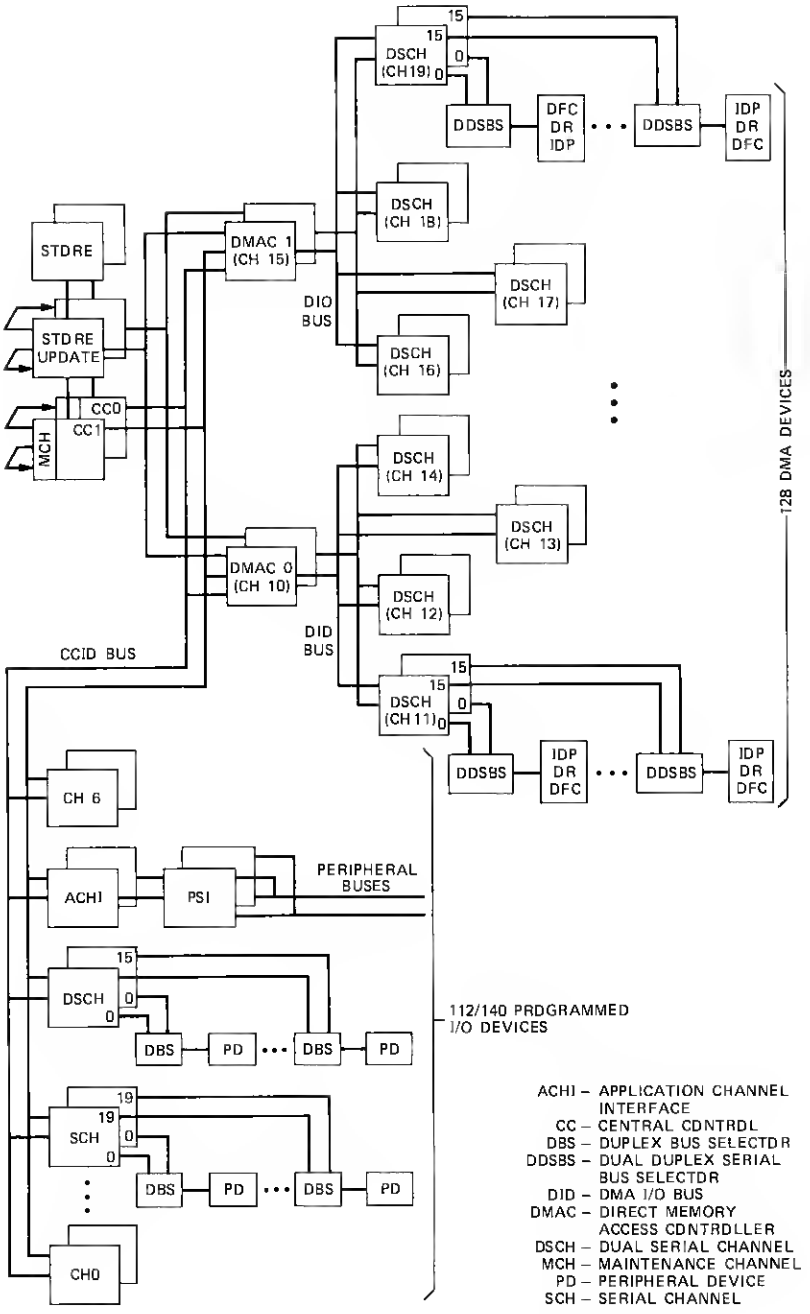


Fig. 4—The 3B20D Processor input/output architecture.

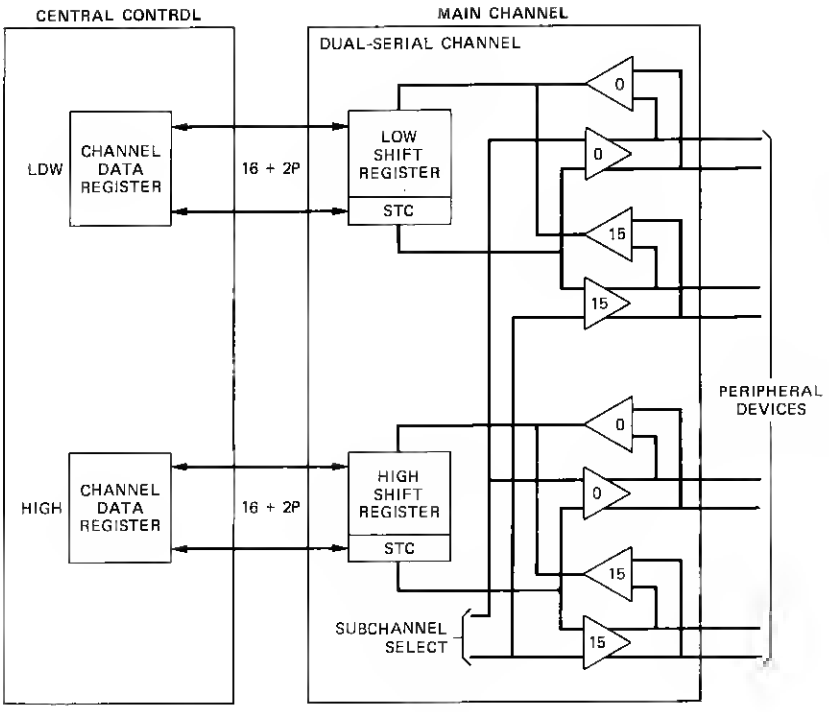


Fig. 5—Dual-serial channel.

the facility for moving data between I/O devices and main memory without having the processor involved in transferring the data. The DMA circuit consists of a DMAC and one to four DSCHs. A DMA can accommodate up to 64 devices, all of which may be active concurrently. The DMAC supports virtual addressing, word or block (16 word) transfers, device-initiated transfers, and multiple jobs for a given device. The latter function allocates a unique segment in memory to each job for a device. This prevents one job from mutilating another job's memory.

A DMA transfer is a two-step operation. Half the time is spent with the channel and peripheral device passing data and half with the channel-DMAC-main store passing data. While the former operation is in progress, the DMAC can be loading or unloading another channel from main store. The DMAC has a hardware priority circuit that gives channel 0 priority over channel 1, which has a higher priority than channel 2, etc. The devices on a given channel are prioritized in the same manner, i.e., device 0 having the highest priority and device 15 the lowest. The channel does not permit interleaving devices on less than a block or word boundary. Once a transfer of a block or word is

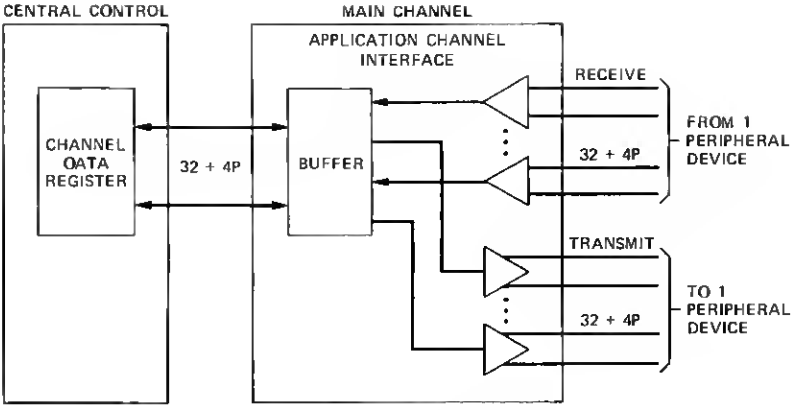


Fig. 6—Application channel interface.

initiated to a device, that transfer must be completed before another device on that channel can be serviced.

A DMA transfer is initiated by the device passing the DMAC the starting virtual address and, optionally, a transfer count. The DMAC translates the address into a physical address as described below. The transfer count is used as a check verifying that the device and DMAC are in agreement as to the number of transfers to take place.

The address-translation process used by the DMAC is the same as that used by the CC with both using the translation-page tables that are stored in main memory. Each page has protection bits defining DMA read/write access capability. The maximum size transfers that can be accomplished with a single address setup is one segment or 64 2K-byte pages. As part of the initialization process for the DMAC, the processor passes a unique page table pointer to the DMA for each of its active devices. The DMAC uses the page table pointer and the virtual address to obtain the desired physical page pointer. As the DMA transfer crosses a page boundary, the DMAC automatically accesses the page table to obtain the next physical page pointer.

After setting up the DMAC the device initiates the transfer by sending a transfer request. The DMAC will ask for the data from the device or send the data to the device in a word (32 bits) or block (16 words) mode. The device then sends another transfer request and the handshaking continues until the entire job is completed.

III. MAINTENANCE FEATURES

The 3B20D places great emphasis on maintenance features. A complete, fully developed package of such features includes self-checking, fault recovery, and diagnostic capabilities.

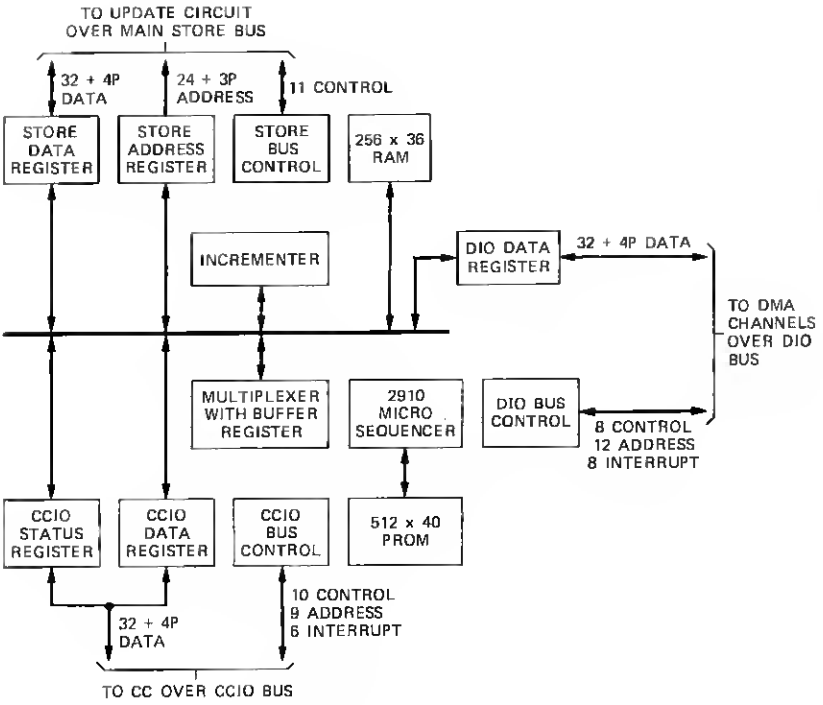


Fig. 7—Direct memory access controller.

3.1 *Self-checking hardware*

Self-checking is implemented in the 3B20D design for concurrent error detection. The maintenance philosophy is to provide a sufficient amount of hardware to enable detection of nearly all service-affecting single hardware faults.[5] To minimize the potential sources of errors in the main memory,[2] single-bit Hamming correction and double-bit error detection are employed. Most software faults such as memory-protection violations, illegal instructions, and out-of-range addresses also are detected. Some of the fault-detection techniques used in the 3B20D Processor are:

(*i*) Parity per byte on data paths throughout the internal CC, memory buses, and peripherals

(*ii*) Single-bit Hamming correction and double-bit error detection on the main store data

(*iii*) Duplicated arithmetic and logic unit and other control logic

(*iv*) "Watchdog" sanity timers for software faults.

Faults detected by the processor check hardware are collected together into a single error register. The action taken for a particular fault depends on its impact on the system. These actions range from

microinterrupts handled in firmware to a stop-and-switch response where control is transferred to the standby half of the duplex processor.

3.2 Hardware fault-recovery features

Fault detection is the first and most important step in realizing a highly reliable system. Almost of equal importance is the rapid recovery by the system. Recovery is achieved by a coordinated combination of hardware, microcode, and software actions. As soon as an error is detected, immediate action takes place to reconfigure the system into an error-free working system. The recovery process involves two steps: reconfiguration and initialization. To facilitate this rapid recovery from system faults, the 3B20D Processor provides the following.

3.2.1 A memory update unit

This unit couples the on-line memory to the off-line memory. The off-line processor's memory is updated on each memory write operation, whether CC or DMA originated, to provide continuous agreement with the on-line memory.

3.2.2 A maintenance channel

This unit directs the off-line processor to initialize and recover when the on-line processor has detected critical error conditions.

3.2.3 Initialization microcode

This nondestructive microcode makes critical recovery decisions when error conditions are detected. This microcode is particularly important if total software sanity is lost.

3.3 Diagnostic features

Hardware has been incorporated into the design of this system to permit a systematic approach for identifying failures via software. This diagnostic software[6] depends heavily on the maintenance channel and its associated circuitry. The primary function of the MCH is the diagnosis of one CPU by the other. The MCH is an autonomous portion of the processor which, under control of the other CPU, can provide information about the state of the machine. The MCH thus exercises the machine at its most basic level by direct access to the microprogram control. The MCH characteristics are like those of the DSCH, and the access protocols are compatible with each other.

IV. MICROCODE FEATURES

The large microstore address space in the 3B20D Processor provides a relatively inexpensive means of specifying complex instructions and

special system functions. These can be added or modified with little difficulty compared to hardwired functions.

4.1 Types of features

Microcode can reside in either PROM or in writeable control store. The following sequences have been implemented in the 3B microcode:

(*i*) Microboot—This sequence of code initializes the processor and loads both the Writable Control Store (WCS) and the first-level bootstrap program from disk. If it is unable to load from the primary disk, it automatically tries to load from the backup disk.

(*ii*) Tapeboot—This sequence of microcode initializes the processor and then copies data from a standard nine-track magnetic tape to the disk. The microboot routine can then be used to initiate program execution.

(*iii*) Basic instruction set—This set of microcode implements the native instruction set and special instructions that are specific to the 3B20D Processor.

(*iv*) External interrupt routine—This sequence is invoked only upon the completion of an instruction with an interrupt pending. The microcode saves the state of the system, then transfers to an interrupt event handler routine.

(*v*) Error interrupt routine—This section of microcode is entered in case of a hardware or software error. The microcode saves the state of the machine and then transfers control to a routine that attempts to recover processing without switching to the other machine. This interrupt can be encountered between any microinstructions.

(*vi*) Memory management trap—This sequence of microcode is entered if the virtual address of the memory fetch cannot be translated directly to a physical address by the address-translation hardware. The microcode reads the segment and page tables of the active process, loads the hardware translation unit with the translation information and then reactivates the memory access.

(*vii*) Maintenance channel instructions—These instructions allow the processor to communicate with its duplex mate via the maintenance channel.

(*viii*) Diagnostic sequences—A section of the WCS is reserved for the diagnostic programs to load and then execute special microcode sequences.

(*ix*) Miscellaneous routines—There are several miscellaneous microcode routines that are used for functions such as to load the writable control store and to load the hardware matcher registers in the utility circuit.

(*x*) Emulation routines—Up to three additional sets of microcode can be loaded that allow the 3B20D to emulate other machines.

4.2 Microcode development

Several software and hardware tools were developed that allowed microcode to be generated and debugged with relative ease. Of principal importance are the MICA[7] bit-sliced assembler and the Micro-level Test Set (MLTS).[6] These tools allowed a laboratory utility computer to be used to assemble microcode, and to control the 3B20D via a link to the MLTS. The facility was used to load writable microstore, access main store, access registers and control sequencing at the microcode level, set breakpoints and provide trace capability.

During the development phase of the 3B20D processor, substantial advantages were realized by the fact that the system was microprogrammed. Major improvements in the instruction set architecture were accommodated by microcode changes. Features have been added to enhance error recovery, to permit transferring bootstrap programs from magnetic tape to disk, and to improve system performance.

V. SUMMARY

The 3B20D CPU is a 32-bit machine with 24-bit addressing. Hardware features have been provided to support a modern general-purpose operating system, e.g., virtual-to-physical address translation. Other features include microprogram implementation, emulation capability, high-speed data cache, high-speed interrupt stack, self-checking circuits, extensive diagnostic access, and high-availability duplex operation.

The standard I/O communication between the CC and the peripherals is by means of a DMA controlling dual-serial channels, capable of transmitting an effective rate of 2M-bytes/s. The DMA can operate in a word-transfer mode or a block mode of 16 words per block. The loose coupling of the channels between the processor and the peripherals permits considerable freedom in expanding a system.

VI. ACKNOWLEDGMENTS

Many individuals have contributed in a significant way to the organization, design, and implementation of the 3B20D central processing unit. The authors would like to particularly acknowledge the contributions of C. A. Borcher and C. W. Hoffner to both the development of the 3B20D Processor and to the content of this paper.

REFERENCES

1. *The AM2900 Family Data Book*, Sunnyvale, California: Advanced Micro Devices, 1979.
2. I. K. Hetherington and P. Kusulas, "3B20D Memory Systems," B.S.T.J., this issue.
3. A. H. Budlong and F. W. Wendland, "3B20D Input/Output System," B.S.T.J., this issue.

4. R. E. Haglund and L. D. Peterson, "3B20D File Memory Systems," B.S.T.J., this issue.
5. R. C. Hansen, R. W. Peterson, and N. O. Whittington, "Fault Detection and Recovery," B.S.T.J., this issue.
6. J. L. Quinn, R. L. Engram, and F. M. Goetz, "Diagnostic Tests and Control Software," B.S.T.J., this issue.
7. J. T. Beckett and S. W. Ng, "A General Purpose Microcode Assembler," Proc. IEEE/COMPSAC, 1978, pp. 84–89.